Σύντομος Οδηγός για το Maxima

Ελληνική ομάδα Maxima

ellmaxima [at] gmail [dot] com

Φεβρουάριος 2017

Εισαγωγή

Το **Maxima** είναι ένα υπολογιστικό μαθηματικό σύστημα (computer algebra system, CAS), γραμμένο στην γλώσσα προγραμματισμού Common Lisp. Βασίστηκε στο προγενέστερο σύστημα Macsyma, που είχε αναπτυχθεί από το Αμερικανικό Πανεπιστήμιο MIT από το 1968 ως το 1982. Το 1998 ο πηγαίος κώδικας (source code) δημοσιοποιήθηκε υπό την ελεύθερη άδεια GNU General Public License και λίγο αργότερα ξεκίνησε η ανάπτυξη του προγράμματος από την επιστημονική κοινότητα. Το **Maxima** σήμερα διατίθεται ως ελεύθερο και δωρεάν λογισμικό σε όλα τα κύρια λειτουργικά συστήματα: GNU/Linux, BSD, MacOS, και Microsoft Windows.

Το **Maxima** υποστηρίζει ένα ευρύ φάσμα συμβολικών μαθηματικών υπολογισμών καλύπτοντας, μεταξύ άλλων: παραγώγιση, ολοκλήρωση, σειρές Taylor, διαφορικές εξισώσεις, μετασχηματισμούς Laplace, πράξεις συνόλων, θεωρία αριθμών, συστήματα γραμμικών εξισώσεων, άλγεβρα πολυωνύμων, διανυσμάτων, τανυστών κ.α.

Η επίσημη ιστοσελίδα του προγράμματος βρίσκεται στην ηλεκτρονική διεύθυνση:

http://maxima.sourceforge.net/

Ακολουθεί μία σύντομη εισαγωγική παρουσίαση χρήσης του προγράμματος, συγκεκριμένα μέσω της πλέον δημοφιλούς διεπαφής **wxMaxima**. Από τις εκατοντάδες εντολές (και αντίστοιχες δυνατότητες) παρουσιάζονται εδώ μόνο ορισμένες βασικές λειτουργίες μέσα από απλά παραδείγματα. Η διαθέσιμη τεκμηρίωση στην Αγγλική γλώσσα είναι εκτενέστατη, και βρίσκεται στην ιστοσελίδα του προγράμματος.

Αριθμητικές πράξεις

Το απλούστερο πράγμα που μπορεί να κάνει κανείς με το Maxima είναι οι συνήθεις αριθμητικές πράξεις, οι οποίες συμβολίζονται ως εξής: **+** πρόσθεση, **-** αφαίρεση, ***** πολλαπλασιασμός, **/** διαίρεση, **^** ή ****** ύψωση σε δύναμη, **!** παραγοντικό.

Όταν εκκινεί η διεπαφή **wxMaxima** εμφανίζεται μία λευκή επιφάνεια, με την γραμμή επιλογών στο άνω μέρος του παραθύρου. Αν δώσουμε:

```
(11-7)*(6/2+5)^2
```

και πιέσουμε **Shift+Enter** (προσοχή, όχι σκέτο **Enter**), το πρόγραμμα θα επιστρέψει τις γραμμές:

```
(%i1) (11-7)*(6/2+5)^2;
(%o1) 256
```

Το γράμμα **i** αντιστοιχεί στη λέξη **input** (στοιχείο εισόδου, δεδομένο), το γράμμα **o** στη λέξη **output** (στοιχείο εξόδου, αποτέλεσμα), ενώ ο δείκτης **1** στην αρίθμηση των εκτελεσμένων πράξεων. Πιέζοντας διαδοχικά το μικρό τρίγωνο πάνω αριστερά, αποκρύπτεται ή επανεμφανίζεται το αποτέλεσμα. Αυτό μπορεί να φανεί χρήσιμο στην περίπτωση που τα ενδιάμεσα αποτελέσματα καταλαμβάνουν σημαντικό χώρο στο παράθυρο εργασίας. Μπορούμε να δώσουμε περισσότερες από μία συνεχόμενες εντολές, τις οποίες χωρίζουμε με το ελληνικό ερωτηματικό `;` (το ελληνικό ερωτηματικό **δεν** είναι το σύμβολο **?**).

Μπορούμε να αναφερθούμε εκ νέου στα δεδομένα εισόδου, χρησιμοποιώντας το σύμβολο **%**. Έτσι, η ακόλουθη εντολή θα αφαιρέσει το **10** από το αμέσως προηγούμενο αποτέλεσμα:

```
(%i2) %-10;
(%o2) 246
```

Αν θέλουμε να αναφερθούμε πάλι στο αποτέλεσμα **256** (επί παραδείγματι για να το διαιρέσουμε με το **8**), μπορούμε να γράψουμε:

```
(%i3) %o1/8,numer;
(%o3) 32
```

Το όρισμα **numer** προστέθηκε για να υπολογιστεί αριθμητικά το αποτέλεσμα (δοκιμάστε ξανά χωρίς αυτό). Το πρόγραμμα γενικά επιστρέφει το αποτέλεσμα μιας διαίρεσης ως ανάγωγο κλάσμα:

```
(%i4) 15/95;
(%o4)  3/19
```

Γενικώτερα, το πρόγραμμα εκτελεί τις ζητούμενες πράξεις κάνοντας ακριβείς συμβολικούς υπολογισμούς: αυτή είναι και η μεγάλη του χρησιμότητα. Αν όμως θέλουμε το αποτέλεσμα σε δεκαδική μορφή, μπορούμε να χρησιμοποιήσουμε την εντολή **float**.

```
(%i5) float(%);
(%o5) 0.1578947368421053
```

Μπορούμε να δηλώσουμε ότι θέλουμε το αποτέλεσμα σε μορφή δεκαδικού εναλλακτικά και ως εξής (το αποτέλεσμα είναι το ίδιο, γι' αυτό παραλείπεται):

```
(%i6) 15/95.0;
(%i7) 15.0/95;
(%i8) 15/95,numer;
```

Το **Maxima** είναι σε θέση να χειριστεί πολύ μεγάλους αριθμούς: για να υπολογίσουμε το **33** παραγοντικό:

```
(%i9) 33!;
(%o9) 8683317618811886495518194401280000000
```

Οι πολύ μεγάλοι αριθμοί δεν είναι πάντα βολικοί στην πλήρη τους μορφή. Με την εντολή **float** μπορούμε να μετατρέψουμε έναν αριθμό σε τυποποιημένη εκθετική μορφή:

```
(%i10) float(%);
(%o10) 8.683317618811885 10^36
```

Όταν το αποτέλεσμα μιας δύναμης είναι άρρητος αριθμός, το πρόγραμμα δεν εκτελεί την πράξη (εκτός αν χρησιμοποιήσουμε τις τεχνικές όπως προηγουμένως με τα κλάσματα) αλλά δίνει το αποτέλεσμα ως δόθηκε:

```
(%i11) (1+sqrt(2))^5;
(%o11) (sqrt(2)+1)^5
```

Αν θέλουμε να αναπτύξουμε συμβολικά την δύναμη, χρησιμοποιήσουμε την εντολή **expand**:

```
(%i12) expand(%);
(%o12) 29*sqrt(2)+41
```

Για προσεγγιστικό υπολογισμό τέλος, όπως είδαμε ανωτέρω:

```
(%i13) float(%);
(%o13) 82.01219330881976
```

Συναρτήσεις και βασικές σταθερές

Μπορούμε να χρησιμοποιήσουμε αρκετές μαθηματικές σταθερές στους υπολογισμούς μας, όπως για παράδειγμα:

%e	ο αριθμός **e** του Euler (**2.71828...**)
%pi	ο αριθμός **π** (**3.14159...**)
%phi	ο «χρυσός» αριθμός **φ** (**1.61803...**)
%i	η φανταστική μονάδα
inf	το **+** άπειρο
minf	το **-** άπειρο

Ορισμένες βασικές συναρτήσεις συμβολίζονται εξής:

sin	ημίτονο
cos	συνημίτονο
tan	εφαπτομένη
cot	συνεφαπτομένη
sec	τέμνουσα
atan	τόξο εφαπτομένης
sqrt	τετραγωνική ρίζα
log	φυσικός λογάριθμος
exp	εκθετική συνάρτηση
abs	απόλυτη τιμή

Ας δούμε κάποιες συναρτήσεις στην πράξη:

```
(%i14) (sin(%pi/2)+cos(%pi/3))*sqrt(81);
(%o14)  27/2
(%i15) exp(%pi*%i);
(%o15) -1
```

Μπορούμε να ορίσουμε κάποιες παραμέτρους που σκοπεύουμε να χρησιμοποιήσουμε στη συνέχεια, με το σύμβολο της διπλής τελείας **:**

```
(%i17) a:3;b:5;
(a)    3
(b)    5
```

Αυτό σημαίνει ότι στα επόμενα, όταν το πρόγραμμα συναντά τα γράμματα **a** και **b**, θα τα αντικαθιστά με τις τιμές **3** και **5**, αντίστοιχα. Ένα παράδειγμα:

```
(%i18) sqrt(b^2-a^2);
(%o18) 4
```

Με ανάλογο τρόπο, με το σύμβολο **:=** ορίζονται οι συναρτήσεις:

```
(%i19) f(x):=x^2-x+1;
(%o19)  f(x):=x^2-x+1
(%i20) f(b)-f(a);
(%o20) 14
```

Σημειώνεται ότι το πρόγραμμα δεν διαθέτει ως έτοιμη συνάρτηση τον δεκαδικό λογάριθμο, οπότε πρέπει να οριστεί ώστε να χρησιμοποιηθεί στα επόμενα:

```
(%i21) log10(x):= log(x)/log(10);
(%o21)  log10(x):=log(x)/log(10)
```

Για να υπολογίσουμε αθροίσματα μιας ακολουθίας χρησιμοποιούμε την εντολή **sum**. Ας υποθέσουμε ότι ζητούμε το άθροισμα των όρων **1/n** όπου το **n** λαμβάνει τις ακέραιες τιμές από **1** ως **100**:

```
(%i22) sum(1/n,n,1,100),numer;
(%o22) 5.18737751763962
```

Τα γινόμενα ακολουθιών υπολογίζονται με την εντολή **product**. Ας δούμε ένα άλλο παράδειγμα:

```
(%i23) product(1+1/n^2,n,1,100),numer;
(%o23) 3.63968229453131
```

Η χρήση του συμβόλου **$** αντί για **;** στο τέλος ενός στοιχείου εισόδου είναι σκόπιμη σε περίπτωση που εκτελούμε μια αλυσίδα πράξεων και δεν επιθυμούμε να εμφανίζονται τα ενδιάμεσα αποτελέσματα. Στο επόμενο παράδειγμα δίνουμε την ακτίνα **r** και το ύψος **h** κυλίνδρου και στο αμέσως επόμενο βήμα υπολογίζουμε τον όγκο του:

```
(%i24) r:2$
(%i25) h:5$
(%i26) V:%pi*r^2*h;
(V)    20 π
```

Θεωρία αριθμών

Η εντολή **factor** αναλύει έναν αριθμό σε γινόμενο πρώτων παραγόντων:

```
(%i27) factor(10!);
(%o27)  2^8 3^4 5^2 7
```

Με την εντολή **ifactors** λαμβάνουμε την πρωτογενή ανάλυση του αριθμού **n** με τη μορφή διατεταγμένων ζευγών:

```
(%i28) ifactors(10!);
(%o28) [[2,8],[3,4],[5,2],[7,1]]
```

Αν δώσουμε ένα ζεύγος αριθμών, η εντολές **quotient** και **remainder** επιστρέφουν το πηλίκο και το υπόλοιπο της διαίρεσης του πρώτου προς τον δεύτερο, αντίστοιχα:

```
(%i29) quotient(138,17);
(%o29) 8
(%i30) remainder(138,17);
(%o30) 2
```

Ας δούμε ένα συμβολικό αλγεβρικό υπολογισμό: ας υποθέσουμε ότι θέλουμε να δείξουμε ότι ο αριθμός **7** διαιρεί την παράσταση $3^{2n+1}+2^{n+2}$. Αρκεί να δείξουμε ότι το υπόλοιπο της διαίρεσης είναι **0**:

```
(%i31) expr:3^(2*n+1)+2^(n+2);
(expr)  3^(2n+1)+2^(n+2)
(%i32) remainder(expr,7);
(%o32) 0
```

Οι εντολές **gcd** και **lcm** δίνουν τον μέγιστο κοινό διαιρέτη και το ελάχιστο κοινό πολλαπλάσιο δύο αριθμών:

```
(%i33) gcd(18,30);
(%o33) 6
(%i34) lcm(18,30);
(%o34) 90
```

Η εντολή **gcd** δέχεται ως όρισμα μόνο ένα ζεύγος αριθμών. Αν όμως θέλουμε να βρούμε τον μέγιστο κοινό διαιρέτη τριών ή περισσότερων αριθμών, μπορούμε να εφαρμόσουμε την ιδιότητα **(a,b,c)=((a,b),c)**. Έτσι:

```
(%i35) gcd(gcd(12,15),18);
(%o35) 3
```

Η εντολή **primep** ελέγχει αν ένας δεδομένος αριθμός είναι ή όχι πρώτος και επιστρέφει την αληθοτιμή (**true** ή **false**):

```
(%i36) primep(33245677);
(%o36) true
```

Για να βρούμε τον αμέσως μεγαλύτερο πρώτο αριθμό ενός δεδομένου αριθμού n, χρησιμοποιούμε την εντολή **next_prime**. Ανάλογη είναι η εντολή **prev_prime** που επιστρέφει τον αμέσως μικρότερο πρώτο:

```
(%i37) next_prime(20);
(%o37) 23
(%i38) prev_prime(20);
(%o38) 19
```

Οι θετικοί ακέραιοι που είναι μικρότεροι από **10** και πρώτοι προς αυτόν είναι οι **1,3,7,9**. Το πλήθος τους είναι ίσο προς **4**. Αυτό μπορεί να υπολογιστεί με την εντολή **totient**:

```
(%i39) totient(10);
(%o39) 4
```

Οι θετικοί ακέραιοι που διαιρούν το **10** είναι οι **1,2,5,10**. Το άθροισμά τους είναι ίσο προς **18**. Αυτό μπορεί να υπολογιστεί με την εντολή **divsum**:

```
(%i40) divsum(10);
(%o40) 18
```

Για να υπολογίσουμε το άθροισμα των **k**–δυνάμεων των διαιρετών ενός φυσικού αριθμού **n**, $\sum_{d|k} d^k$, δίνουμε στην εντολή **divsum** ως όρισμα ένα ζεύγος αριθμών, όπου ο πρώτος δηλώνει τον αριθμό **n** και ο δεύτερος την επιθυμητή δύναμη **k**:

```
(%i41) divsum(10,2);
(%o41) 130
```

Η εντολή **binomial** υπολογίζει διωνυμικούς συντελεστές $\binom{n}{m} = \frac{n!}{m!(n-m)!}$:

```
(%i42) binomial(10,3);
(%o42) 120
```

Σύνολα

Ένα σύνολο δηλώνεται με την εντολή **set**, ή με άγκιστρα, αναγράφοντας τα στοιχεία του συνόλου. Αν δεν δώσουμε κανένα στοιχείο, αναφερόμαστε στο κενό σύνολο:

```
(%i43) a:set(1,2,3,{1},A);
(a)    {1,2,3,{1},A}
(%i44) b:{3,4,5,A,B};
(b)    {3,4,5,A,B}
```

Για να βρούμε την ένωση (**union**) ή την τομή (**intersection**) των **a** και **b**:

```
(%i45) c:union(a,b);
(c)    {1,2,3,4,5,{1},A,B}
```

```
(%i46) d:intersection(a,b);
(d)    {3,A}
```

Η διαφορά δύο συνόλων **A-B** ή **A\B** δηλώνεται με την εντολή **setdifference**:

```
(%i47) dif:setdifference(a,b);
(dif)  {1,2,{1}}
```

Με τις εντολές **powerset** και **cartesian_product** βρίσκουμε το δυναμοσύνολο ενός συνόλου και το καρτεσιανό γινόμενο δύο συνόλων αντίστοιχα:

```
(%i48) powerset(dif);
(%o48) {{},{1},{1,2},{1,2,{1}},{1,{1}},{2},{2,{1}},{{1}}}
(%i49) cartesian_product({x,y,z},dif);
(%o49) {[x,1],[x,2],[x,{1}],[y,1],[y,2],[y,{1}],[z,1],[z,2],[z,{1}]}
```

Η εντολή **cardinality** δίνει τον πληθάριθμο ενός συνόλου (το πλήθος των στοιχείων του):

```
(%i50) cardinality(c);
(%o50) 8
```

Πολυώνυμα και ρητές παραστάσεις

Ο τελεστής **factor** αναλύει ένα πολυώνυμο σε γινόμενο πρώτων παραγόντων. Η ανάλυση αυτή γίνεται στο σύνολο των πολυωνύμων με ακέραιους συντελεστές. Αυτό σημαίνει ότι μπορούμε να κάνουμε μία ανάλυση της μορφής: $(x^2+x-6)=(x+3)(x-2)$ αλλά όχι μία της μορφής: $(x^2-2)=(x-\sqrt{2})(x+\sqrt{2})$ ή $(x^2+1)=(x-i)(x+i)$. Για να παραγοντοποιήσουμε την παράσταση $3x^3+11x^2+8x-4$:

```
(%i51) factor(3*x^3+11*x^2+8*x-4);
(%o51) (x+2)^2(3x-1)
```

Ο τελεστής **expand**, αντίθετα, αναπτύσσει δυνάμεις και γινόμενα πολυωνύμων:

```
(%i52) expand((x+2)^2*(3*x-1));
(%o52) 3x^3+11x^2+8x-4
```

Μπορούμε να αντικαταστήσουμε κάποιες μεταβλητές με άλλες, ή και με αριθμητικές τιμές, ως εξής:

```
(%i53) expand((x+y+1)^2);
(%o53) y^2+2xy+2y+x^2+2x+1
(%i54) %,x=q,y=1;
(%o54) q^2+4q+4
```

Εδώ να σημειώσουμε το εξής: όταν δηλώνουμε την αντικατάσταση μιας μεταβλητής με μία παράμετρο (όπως εδώ την **q**) υπάρχει ο κίνδυνος να χρησιμοποιούμε ένα όνομα παραμέτρου που είχε οριστεί ξανά προηγουμένως, και συνεπώς έχει ήδη κάποια καθορισμένη τιμή, ή έκφραση. Αν θέλουμε να την ορίσουμε ξανά, μπορούμε να την «καθαρίσουμε» με την εντολή **kill**:

```
(%i55) kill(q);
(%o55) done
```

Όποτε δεν είμαστε βέβαιοι, οι εντολές **values** και **functions** μας ενημερώνουν για τις παραμέτρους και τις συναρτήσεις που έχουν ήδη καθορισμένη τιμή, ώστε να μην τις χρησιμοποιήσουμε λανθασμένα.

Με τη χρήση της εντολής **solve** λύνονται αλγεβρικές εξισώσεις. Στο όρισμα πρέπει να δωθεί η εξίσωση (αν παραλειφθεί το **=0**, το Maxima θα το θεωρήσει δεδομένο) και να καθοριστεί και ο άγνωστος της εξίσωσης. Έτσι:

```
(%i56) solve(x^2-4=0,x);
(%o56) [x=-2,x=2]
```

Το **Maxima** επιστρέφει τη λύση μιας εξίσωσης, μόνον εφόσον η τελευταία είναι επιλύσιμη με ριζικά. Είναι γνωστό ότι οι πολυωνυμικές εξισώσεις βαθμού **5** ή μεγαλύτερου δεν έχουν γενικά τέτοια «κλειστή» λύση. Σε μια τέτοια περίπτωση το **Maxima** εξετάζει την πιθανή ανάλυση του πολυωνύμου σε γινόμενο παραγόντων (σύμφωνα με όσα συζητήθηκαν ήδη για την εντολή **factor**) και στη συνέχεια λύνει χωριστά τις όποιες εξισώσεις προκύπτουν, δίνοντας τις ακριβείς τιμές των ριζών, όπου υπάρχουν. Αν, για παράδειγμα, ένα πολυώνυμο βαθμού **7** αναλύεται σε γινόμενο δύο αναγώγων πολυωνύμων βαθμού **4** και **3** αντίστοιχα, θα πάρουμε την πλήρη λύση, αφού οι βαθμοί των παραγόντων είναι μικρότεροι του **5**, που σημαίνει ότι οι δύο επιμέρους εξισώσεις λύνονται αλγεβρικά. Αν όμως αναλύεται σε γινόμενο τριών αναγώγων πολυωνύμων, ενός βαθμού **5** και δύο πρωτοβάθμιων, θα πάρουμε μόνο **2** ρίζες, αφού ένα ανάγωγο πολυώνυμο βαθμού **5** δεν έχει κλειστή λύση. Ένα τέτοιο παράδειγμα είναι η εξίσωση: $x^7-2x^6-4x^5+13x^4-12x^3+5x^2-3x+2=0$.

```
(%i57) solve(x^7-2*x^6-4*x^5+13*x^4-12*x^3+5*x^2-3*x+2);
(%o57) [x=1,x=2, 0=x^5+x^4-3x^3+2x^2+1 ]
```

Το **Maxima** μπορεί να λύσει και παραμετρικές εξισώσεις:

```
(%i59) kill(a)$ solve(x^2-a*x+1,x);
```

(%o59) $[x=-\frac{\sqrt{a^2-4}-a}{2},x=\frac{\sqrt{a^2-4}+a}{2}]$

Η εντολή **nroots** υπολογίζει το πλήθος των πραγματικών ριζών μιας πολυωνυμικής εξίσωσης σε ένα δεδομένο διάστημα της μορφής **(,]**. Ας υπολογίσουμε το πλήθος των πραγματικών λύσεων μιας εξίσωσηςστο διάστημα **(-5,3]**:

```
(%i60) p:x^7-3*x^4+1;
(p)    x^7-3x^4+1
(%i61) nroots(p,-5,3);
(%o61) 3
```

Το **Maxima** επιστρέφει τόσο πραγματικές, όσο και μιγαδικές ρίζες:

```
(%i62) solve(x^2+2*x+5,x);
(%o62) [x=-2 %i-1,x=2 %i-1]
```

Στο σημείο αυτό είναι χρήσιμο να υπογραμμιστεί η μεγάλη σημασία της ορθής σύνταξης από την πλευρά του χρήστη. Αν δώσουμε:

```
(%i64) solve(x^2-a,x); a:4;
(%o63) [x=-sqrt(a),x=sqrt(a)]
(a)    4
```

Δηλαδή το πρόγραμμα αντιλαμβάνεται τις δύο εντολές της γραμμής **(%i64)** ανεξάρτητα, επειδή χωρίζονται με το ερωτηματικό. Εκτελεί λοιπόν την πρώτη, λύνει δηλαδή την εξίσωση ως προς **x**, και στη συνέχεια αποδίδει στην μεταβλητή **a** την τιμή **4**. Αν αλλάξουμε ελαφρά τη σύνταξη:

```
(%i65) solve(x^2-b,x), b:4;
(%o65) [x=-2,x=2]
```

Εδώ οι εντολές χωρίστηκαν με κόμμα αντί για ερωτηματικό, οπότε το πρόγραμμα τις αντιλήφθηκε ως μία ενιαία εντολή. Επέστρεψε συνεπώς την λύση της εξίσωσης όπου η παράμετρος **b** έλαβε την τιμή **4**. Αν αλλάξουμε όμως τη σειρά των εντολών:

```
(%i67) c:4; solve(x^2-c,x);
(c)    4
(%o67) [x=-2,x=2]
```

Εδώ το πρόγραμμα δέχθηκε (και εκτέλεσε) ως πρώτη εντολή την απόδοση της τιμής **4** στην μεταβλητή **c**, την οποία διατηρεί εφεξής στη μνήμη, και στη συνέχεια έλυσε την εξίσωση, στην οποία συνάντησε την μεταβλητή **c**, οπότε και την αντικατέστησε με την ήδη καθορισμένη τιμή της. Τέλος ας δούμε άλλη μια σύνταξη:

```
(%i68) d:4, solve(x^2-d,x);
(d)    4
```

Εδώ φαίνεται καθαρά το αποτέλεσμα της ιδιαίτερης σύνταξης. Η εντολή **d:4** εκτελέστηκε, χρησιμοποιώντας ως επεξήγηση ό,τι ακολούθησε μετά το κόμμα. Στην συγκεκριμένη περίπτωση η επεξήγηση δεν επηρέασε το αποτέλεσμα.

Σχετική με τα προηγούμενα είναι η εντολή **ev**, η οποία δέχεται ως όρισμα μία σειρά υπολογισμών και τις εκτελεί σε ένα «στεγανό» περιβάλλον. Αν εκεί αποδοθούν τιμές σε κάποιες μεταβλητές, αυτές θα ξεχαστούν στη συνέχεια. Ομοίως, αν εκεί χρησιμοποιηθούν κάποιες μεταβλητές με ήδη καθορισμένη τιμή, δεν αντικαθίστανται. Για παράδειγμα:

```
(%i69) ev(solve(a1*x^2+b1*x+c1,x),a1:2,b1:3,c1:-2);
```

(%o69) $[x=-2,\ x=\frac{1}{2}\ ]$

```
(%i70) a1;
(%o70) a1
```

Το πρόγραμμα εκτέλεσε τους υπολογισμούς της γραμμής **%i70** μέσα στο περιβάλλον **ev** και στη συνέχεια «ξέχασε» τις τιμές των μεταβλητών. Υπενθυμίζεται ότι για να «καθαρίσουμε» μια μεταβλητή από την τιμή που της έχει αποδοθεί, χρησιμοποιούμε την εντολή **kill**. Στο όρισμα περιλαμβάνουμε όσες μεταβλητές θέλουμε, ή και όλες αν γράψουμε **kill(all)**.

Η εντολή **partfrac** αναλύει μία ρητή παράσταση σε μερικά κλάσματα. Για να αναλύσουμε την παράσταση $a=\frac{x}{x^2-5x+6}$

```
(%i71) a:x/(x^2-5*x+6);
```

(a) $\frac{x}{x^2-5x+6}$

```
(%i72) partfrac(a,x);
```

(%o72) $\frac{3}{x-3}-\frac{2}{x-2}$

Με την εντολή **ratsimp** μπορούμε να κάνουμε το αντίστροφο. Ο τελεστής αυτός δρα σε μία ρητή παράσταση επιστρέφοντας ένα ενιαίο ρητό κλάσμα:

```
(%i73) ratsimp(%);
```
$$\text{(\%o73)}\quad \frac{x}{x^2-5x+6}$$

Η εντολή **funcsolve** επιχειρεί να λύσει μια αναδρομική εξίσωση, εφόσον υπάρχει πολυωνυμική ή ρητή λύση. Λύνουμε ως εξής την ακόλουθη εξίσωση:

$$(n+1)f(n)-\frac{n+3}{n+1}f(n+1)=\frac{n-1}{n+2}$$

```
(%i75) kill(f)$ eqn:(n+1)*f(n)-(n+3)*f(n+1)/(n+1)=(n-1)/(n+2);
```
$$\text{(eqn)}\quad (n+1)f(n)-\frac{(n+3)f(n+1)}{n+1}=\frac{n-1}{n+2}$$
```
(%i76) funcsolve(eqn,f(n));
solve: dependent equations eliminated: (4 3)
```
$$\text{(\%o76)}\quad f(n)=\frac{n}{(n+1)(n+2)}$$

Η εντολή **gcd** λειτουργεί, πέρα από θετικούς ακέραιους, και στην περίπτωση πολυωνύμων, δίνοντας τον μέγιστο κοινό διαιρέτη τους. Με την εντολή **ezgcd**, η οποία βασίζεται σε διαφορετικό αλγόριθμο, δίνουμε μία **n**-άδα πολυωνύμων και το πρόγραμμα επιστρέφει μία **(n+1)**-άδα, όπου το πρώτο στοιχείο είναι ο μέγιστος κοινός διαιρέτης των **n** πολυωνύμων και τα υπόλοιπα **n** στοιχεία είναι οι λόγοι των πολυωνύμων προς τον μέγιστο κοινό διαιρέτη. Ένα παράδειγμα:

```
(%i77) ezgcd(x^2+3*x+2,x+1);
(%o77) [x+1,x+2,1]
```

Δηλαδή πήραμε τα 3 στοιχεία $\gcd(x^2+3x+2,x+1)=x+1$, $\frac{x^2+3x+2}{x+1}=x+2$, $\frac{x+1}{x+1}=1$.

Ιδιαίτερο ενδιαφέρον παρουσιάζει η εντολή **eliminate**, με την οποία μπορούμε να κάνουμε απαλοιφή μιας ή περισσότερων μεταβλητών από ένα σύστημα εξισώσεων. Ας υποθέσουμε ότι θέλουμε να βρούμε την καμπύλη κατά την οποία τέμνονται οι επιφάνειες $x^2+y^2+z^2=3$ και $xy+yz+zx=-1$:

```
(%i79) a:x^2+y^2+z^2-3;b:x*y+y*z+z*x+1;
```
$$\text{(a)}\quad z^2+y^2+x^2-3$$
$$\text{(b)}\quad yz+xz+xy+1$$
```
(%i80) eliminate([a,b],[z]);
```
$$\text{(\%o80)}\quad [y^4+2xy^3+(3x^2-3)y^2+(2x^3-4x)y+x^4-3x^2+1]$$

Συστήματα

Για να λύσουμε ένα σύστημα εξισώσεων, δίνουμε στην εντολή **solve** ένα όρισμα που αποτελείται από δύο ζεύγη αγκυλών. Στο πρώτο τοποθετούμε τις εξισώσεις, χωρισμένες με κόμμα, και στο δεύτερο τους αγνώστους, επίσης χωρισμένους με κόμμα. Αν οι εξισώσεις έχουν μόνο πρώτο μέλος, το δεύτερο μέλος τίθεται αυτόματα μηδέν. Για το σύστημα:

$$x-2y=14$$
$$x+3y=9$$

```
(%i81) solve([x-2*y=14,x+3*y=9],[x, y]);
(%o81) [[x=12,y=-1]]
```

Για το παραμετρικό σύστημα:

$$\begin{aligned} x+ay&=1\\ bx-3y&=2 \end{aligned}$$

```
(%i83) kill(a,b)$ solve([x+a*y=1,b*x-3*y=2],[x,y]);
```

(%o83) $[[x=\frac{2a+3}{ab+3},y=\frac{b-2}{ab+3}]]$

Ας λύσουμε και το μη-γραμμικό σύστημα:

$$\begin{aligned} x+yz&=2\\ y-xz&=0\\ x+y&=2 \end{aligned}$$

```
(%i84) solve([x+y*z=2,y-x*z=0,x+y=2],[x,y,z]);
(%o84) [[x=1,y=1,z=1],[x=2,y=0,z=0]]
```

Όριο, παράγωγος και ολοκλήρωμα

Τα όρια υπολογίζονται με τον τελεστή **limit**. Το όρισμα περιλαμβάνει τη συνάρτηση, την ανεξάρτητη μεταβλητή και το σημείο στο οποίο τείνει η ανεξάρτητη μεταβλητή. Ας δούμε τον υπολογισμό ενός γνωστού ορίου:

```
(%i85) f(x):=sin(x)/x;
```

(%o85) $f(x):=\frac{\sin x}{x}$

```
(%i86) limit(f(x),x,0);
(%o86) 1
```

Για το όριο μιας συνάρτησης όταν $x\to 0^+$ δηλώνουμε πρώτα την απαίτηση **x>0** με την εντολή **assume**. Για παράδειγμα:

```
(%i87) assume(x>0);
(%o87) [x>0]
(%i88) limit(x/abs(x),x,0);
(%o88) 1
```

Ένας άλλος τρόπος να βρούμε πλευρικά όρια είναι να δηλώσουμε τα «αριστερά» και «δεξιά» του αριθμού με **minus** και **plus**. Πρώτα όμως πρέπει να αναιρέσουμε την απαίτηση **x>0** που δώσαμε προηγουμένως, με την εντολή **forget**:

```
(%i89) forget(x>0);
(%o89) [x>0]
(%i90) limit(x/abs(x),x,0,plus);
(%o90) 1
(%i91) limit(x/abs(x),x,0,minus);
(%o91) -1
```

Για το γνωστό όριο:

$$\lim_{x\to\infty}\left(1+\frac{1}{x}\right)^x=e$$

```
(%i92) h(x):=(1+1/x)^x;
```
$$(\%o92)\quad h(x):=\left(1+\frac{1}{x}\right)^{x}$$
```
(%i93) limit(h(x),x,inf);
(%o93) %e
```

Η παράγωγος συμβολίζεται με τον τελεστή **diff**. Το όρισμα περιλαμβάνει την συνάρτηση, την ανεξάρτητη μεταβλητή και την τάξη παραγώγισης, η οποία αν παραληφθεί, τίθεται αυτόματα ίση με **1**.

```
(%i94) f:x^x;
```
$$(f)\quad x^{x}$$
```
(%i95) diff(f,x);
```
$$(\%o95)\quad x^{x}\left(\log(x)+1\right)$$

Αν θέλουμε να εξετάσουμε αν η συνάρτηση $y=e^{-t^2}$ αποτελεί λύση της διαφορικής εξίσωσης $y''+2xy'+2y=0$, γράφουμε:

```
(%i96) y:exp(-t^2);
```
$$(y)\quad \%e^{-t^2}$$
```
(%i97) diff(y,t,2)+2*t*diff(y,t)+2*y;
(%o97) 0
```

Το πρόθεμα ' (τόνος) πριν από την παράγωγο **diff(y,x)** δίνει την εντολή στο πρόγραμμα να μην παραγωγίσει το **y** ως προς **x**, αλλά να χειριστεί την παράγωγο συμβολικά ως **dy/dx**.

```
(%i98) 'diff(x^2-x+1,x);
```
$$(\%o98)\quad \frac{d}{dx}\left(x^2-x+1\right)$$

Ο τελεστής **integrate** χρησιμοποιείται για τον υπολογισμό ολοκληρωμάτων. Το όρισμα περιλαμβάνει 4 θέσεις: τη συνάρτηση, την ανεξάρτητη μεταβλητή και τα 2 άκρα του διαστήματος στο οποίο γίνεται η ολοκλήρωση. Αν τα 2 τελευταία παραληφθούν, η ολοκλήρωση γίνεται αόριστα. Αν το πρόγραμμα δεν είναι σε θέση να υπολογίσει το αόριστο ολοκλήρωμα, επιστρέφει την πληροφορία εισόδου. Σημειώνεται ότι το πρόγραμμα δεν δίνει τη σταθερά **c** στο αποτέλεσμα της αόριστης ολοκλήρωσης. Για το ορισμένο ολοκλήρωμα:

$$\int_1^2 \frac{1}{x}dx=\log 2$$

```
(%i99) integrate(1/x,x,1,2);
(%o99) log(2)
```

Για το αόριστο ολοκλήρωμα:

$$\int \frac{x}{x^4+4}dx=\frac{1}{4}\operatorname{atan}\left(\frac{x^2}{2}\right)+c$$

```
(%i100) integrate(x/(x^4+4),x);
```
$$(\%o100)\quad \frac{\operatorname{atan}\left(\frac{x^2}{2}\right)}{4}$$

Το **Maxima** υπολογίζει και μη-γνήσια (γενικευμένα) ολοκληρώματα. Για τον υπολογισμό του $\int_0^{+\infty} e^{-x^2}\,dx$ γράφουμε:

```
(%i101) integrate(exp(-x^2),x,0,inf);
(%o101)  √π/2
```

Συνήθεις διαφορικές εξισώσεις

Η εντολή **ode2** λύνει συνήθεις διαφορικές εξισώσεις πρώτης ή δεύτερης τάξης. Το όρισμα αποτελείται από **3** στοιχεία: την διαφορική εξίσωση, την εξαρτημένη μεταβλητή και την ανεξάρτητη μεταβλητή. Στο αποτέλεσμα το **Maxima** συμβολίζει με **%c** τη σταθερά (για τάξη-**1**) και με **%k1**, **%k2** τις σταθερές (για τάξη-**2**). Οι μέθοδοι που εφαρμόζει το Maxima είναι: γραμμικές, χωριζόμενων μεταβλητών, ακριβείς, ομογενείς, Bernoulli για τάξη-**1** και σταθερών συντελεστών, ακριβείς, γραμμικές ομογενείς που μετασχηματίζονται σε σταθερών συντελεστών, Euler για τάξη-**2**, καθώς και μερικές ακόμη κατηγορίες. Με **'diff(y,x)** συμβολίζουμε την παράγωγο του **y** ως προς **x**, **dy/dx** ώστε να εισαχθεί στην εξίσωση. Για παράδειγμα οι εξισώσεις $\frac{dy}{dx}-\frac{x}{y}=0$ και $y''-3y'+2y=0$ λύνονται ως εξής:

```
(%i103) kill(x,y)$ ode2('diff(y,x)-x/y=0,y,x);
(%o103)  y^2/2 = x^2/2+%c
(%i104) ode2('diff(y,x,2)-3*'diff(y,x)+2*y=0,y,x);
(%o104)  y=%k1%e^(2x)+%k2%e^x
```

Γραμμική άλγεβρα

Ας ορίσουμε ένα μητρώο (πίνακα) στο **Maxima**, με την εντολή **entermatrix**, καθορίζοντας την διάστασή του:

```
(%i105) m:entermatrix(2,2);
Is the matrix 1. Diagonal 2. Symmetric 3. Antisymmetric 4. General
Answer 1, 2, 3 or 4: 4;
```

Δηλώνουμε με τον αντίστοιχο αριθμό αν θα ορίσουμε διαγώνιο, συμμετρικό, αντισυμμετρικό ή τυχαίο μητρώο: ας δώσουμε την απάντηση **4.** Στη συνέχεια το **Maxima** μας ζητάει τα στοιχεία του μητρώου, αρχίζοντας από την πρώτη γραμμή (row) μέχρι να καλύψουμε όλες τις στήλες (columns):

```
Row 1 Column 1: 1;
Row 1 Column 2: -2;
Row 2 Column 1: -3;
Row 2 Column 2: 7;
Matrix entered.
(%o105)  [ 1   -2 ]
         [ -3   7 ]
```

Το **Maxima** δείχνει το μητρώο που δηλώθηκε. Με τις εντολές **transpose**, **determinant** και **invert**, βρίσκουμε το ανάστροφο, την ορίζουσα και το αντίστροφο.

```
(%i106) transpose(m);
(%o106) [ 1  -3 ]
        [ -2  7 ]
(%i107) determinant(m);
(%o107) 1
(%i108) invert(m);
(%o108) [ 7  2 ]
        [ 3  1 ]
```

Ο πολλαπλασιασμός μητρώων δηλώνεται με την τελεία. Μπορούμε να ορίσουμε 2 ή περισσότερα μητρώα (ακολουθώντας την ανωτέρω διαδικασία) και να εκτελέσουμε πράξεις με αυτά. Σημειώνεται ότι η πρόσθεση μητρώου και αριθμού προσθέτει τον αριθμό σε κάθε στοιχείο του μητρώου.

Με τις εντολές **eigenvalues** και **eigenvectors** βρίσκουμε ιδιοτιμές και ιδιοδιανύσματα. Όταν εφαρμοστεί η πρώτη, το πρόγραμμα εξάγει 2 **n**-άδες, όπου **n** η διάσταση του τετραγωνικού μητρώου. Η πρώτη **n**-άδα δηλώνει τις αριθμητικές τιμές των ιδιοτιμών ενώ η δεύτερη την πολλαπλότητα της αντίστοιχης ιδιοτιμής.

```
(%i109) eigenvalues(m);
(%o109) [4-sqrt(15),sqrt(15)+4],[1,1]
```

Γραφήματα

Το **Maxima** επιτρέπει την εξαγωγή γραφημάτων 2 ή 3 διαστάσεων, με τις εντολές **plot2d** και **plot3d**. Μέσα στην εντολή περιλαμβάνουμε το τμήμα του **x**-άξονα που θα σχεδιαστεί.

```
(%i110) plot2d(x^2-x+3,[x,-10,10]);
```

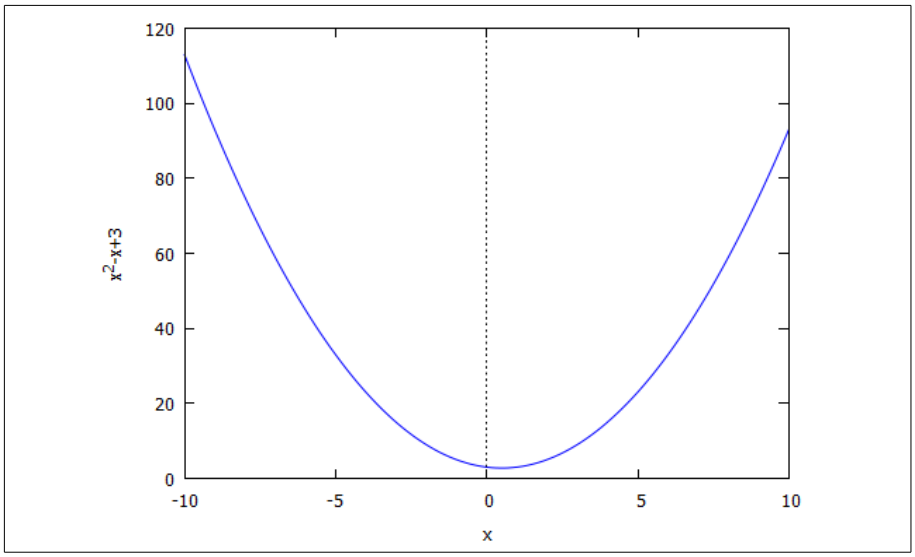
και με **3** διαφορετικές συναρτήσεις στο ίδιο γράφημα:

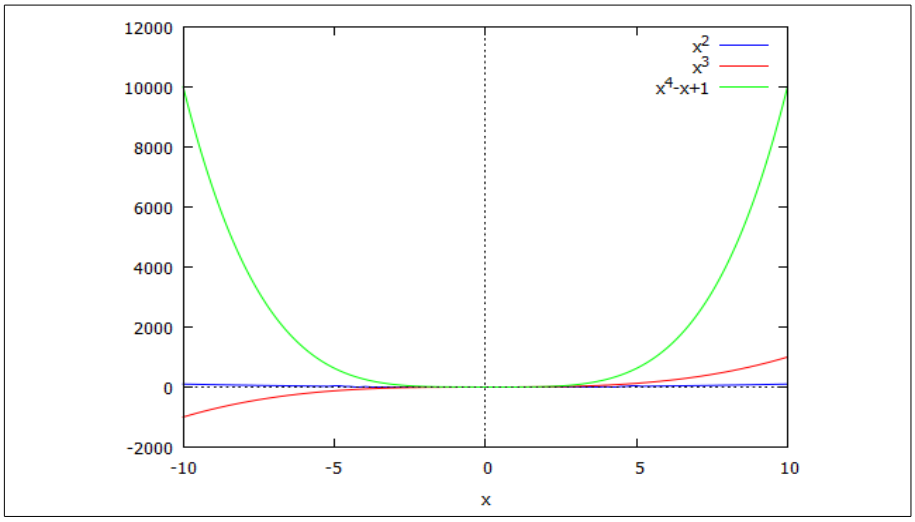
```
(%i111) plot2d([x^2,x^3,x^4-x+1],[x,-10,10]);
```

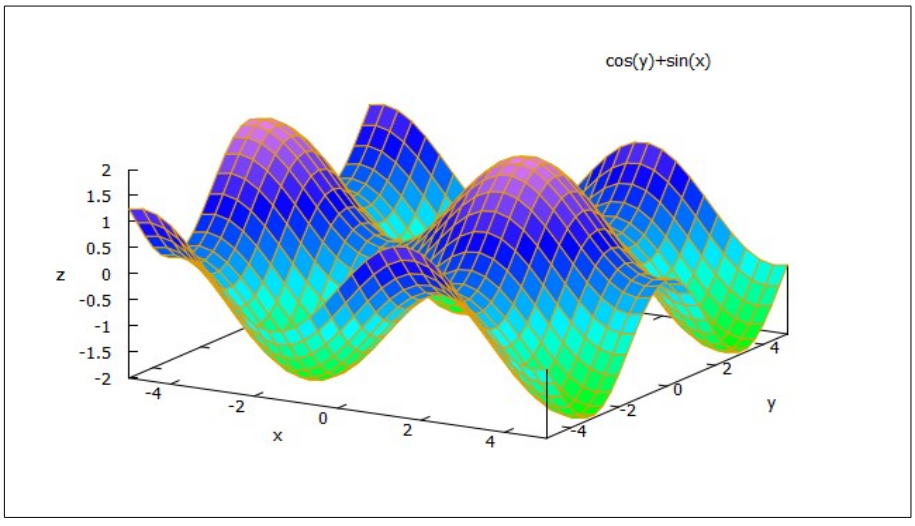
Για γράφημα σε 3 διαστάσεις:

```
(%i112) f(x,y):=sin(x)+cos(y);
(%i113) plot3d(f(x,y),[x,-5,5],[y,-5,5]);
```

Τα 3 γραφήματα φαίνονται στην επόμενη σελίδα.


120
100
80
60
40
20
0
-10
-5
0
5
10
x
x2-x+3


12000
10000
8000
6000
4000
2000
0
-2000
-10
-5
0
5
10
x
x2
x3
x4-x+1


cos(y)+sin(x)
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
z
-4
-2
0
2
4
x
-4
-2
0
2
4
y